In Proceedings of the 4th CSI Computer Conference (CSICC'98), Jan 1999

کوتاه‌ترین مسیر تقریبی در سطوح نامنظم مثلث‌بندی شده‌ی وزن‌دار

محمّد قدسی ghodsi@sharif.ac.ir
یاشار گنجعلی ganjali@ce.sharif.ac.ir

دانشکده‌ی مهندسی کامپیوتر، دانشگاه صنعتی شریف

چکیده

پیدا کردن کوتاه‌ترین مسیر از اساسی‌ترین مسئله‌های مطرح شده در هندسه‌ی محاسباتی و شاخه‌های دیگری مانند الگوریتم‌های گراف، سیستم‌های اطلاعات جغرافیایی (GIS) و برنامه‌ریزی حرکت [۱] در روبات‌هامی‌باشد. کاربردهای بسیار زیاد و متنوع این مسایل، خصوصاً در فضای ۳ـ بعدی، اهمیت فوق‌العاده‌ای به آن بخشیده است. در این مقاله ابتدا به معرفی الگوریتم تقریبی نسبتاً ساده‌ای برای به‌دست آوردن کوتاه‌ترین مسیر در سطوح نامنظم مثلث بندی شده‌ی (TIN [۲]) وزن دار می‌پردازیم. سپس الگوریتم تقریبی جدیدی را ارایه خواهیم داد که می‌توان از آن برای بهبود بخشیدن مسیر تقریبی حاصل از الگوریتم اولیه استفاده نمود. از ویژگی‌های مهم این الگوریتم، زمان اجرای خوب آن نسبت به سایر الگوریتم‌های تقریبی یافتن کوتاه‌ترین مسیر است و نیز این که می‌توان با اجرای متوالی آن مسیر به‌دست آمده را بهبود بخشید.

کلیدواژه‌ها: کوتاه‌ترین مسیر، سطوح نامنظم مثلث‌بندی شده، هندسه‌ی محاسباتی.

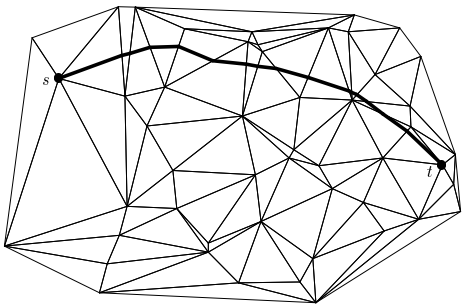
۱ مقدمه

یافتن کوتاه‌ترین مسیر بر روی یک سطح، کاربردهای فراوانی در هندسه‌ی محاسباتی دارد. روش‌های مختلفی برای مدل سازی سطوح مختلف وجود دارد که از آن جمله می‌توان به شبکه‌های منظم مربعی [۳]، مدل خطوط میزان و سطوح نامنظم مثلث بندی شده یا «تین» اشاره کرد (شکل ۱).

در مدل «تین» سطح مورد نظر را با سطح جدیدی که هر یک از وجه‌های آن یک مثلث است، تقریب می‌زنیم، با این شرط که رأس‌های هیچ مثلثی روی اضلاع مثلث دیگری قرار نگیرند. با در نظر داشتن این شرط می‌توان سطح مورد نظر را به‌صورت دل‌خواه مثلث‌بندی کرد. ساختار گسسته‌ی تین و در عین حال انعطاف‌پذیری آن برای نگه‌داری سطوح مختلف، آن را مخصوصاً برای این گونه از مسئله بسیار مناسب می‌سازد.

گونه‌های مختلفی از مسئله‌ی کوتاه‌ترین مسیر بر روی تین تعریف شده است. طبیعی‌ترین گونه‌ی این مسئله یافتن کوتاه‌ترین مسیر بین دو نقطه بر روی سطح یک چند وجهی می‌باشد، به‌طوری که این مسیر به‌طور کامل بر روی سطح واقع شود. به‌عبارت دیگر سطح مثلث‌بندی شده‌ی P که لزوماً محدب هم نیست، و دو نقطه‌ی s و t متعلق به این سطح را در نظر می‌گیریم. برای سادگی فرض می‌کنیم که s و t هر یک رأس یکی از مثلث‌های تین باشند. یک مسیر بین s و t دنباله‌ای متوالی از پاره‌خط هایی است که از s شروع و به t ختم می‌شوند و در عین حال همگی بر روی سطح P قرار دارند. از این پس، کوتاه‌ترین مسیر بین s و t را با $\pi(s,t)$ و مسیر تقریبی بین s و t را با $\pi'(s,t)$ و نیز تعداد رأس‌های تین را با n نمایش می‌دهیم.

[۱] Motion Planning
[۲] Triangulated Irregular Networks
[۳] Regular Square Grid



شکل ۱: یک تین نمونه و کوتاه‌ترین مسیر از s به t.

الگوریتم Hwang و بقیه پیش‌پردازشی از $O(n^{۶}\log n)$ روی سطح مورد نظر انجام می‌دهد و سپس با پیچیدگی زمانی $O(k+\log n)$ کوتاه‌ترین مسیر بین دو رأس مشخص را تعیین می‌کند، که k تعداد یال‌هایی است که کوتاه‌ترین مسیر آن‌ها را قطع می‌نماید [۱]. الگوریتم Sharir و Schorr در حالتی که چند وجهی مورد نظر محدب باشد، با پیچیدگی زمانی $O(n^{۳}\log n)$ کلیه‌ی کوتاه‌ترین مسیرها را بین یک رأس با بقیه‌ی رأس‌ها به دست می‌آورد [۳]. بهترین الگوریتم شناخته شده برای این مسئله، چه در حالت محدب و چه در حالت غیر محدب، الگوریتم مربوط به Chen و Han است[۲]. این الگوریتم که به نوعی تغییر یافته‌ی الگوریتم «دایسترای پیوسته[۴]» می‌باشد به فضایی از $\Theta(n)$ نیاز دارد و از نظر زمانی از $O(n^{۲})$ است. در این الگوریتم حرکت از رأس مبدأ شروع می‌شود و وجه‌های همسایه یکی پس از دیگری طوری حول محور اتصالشان دوران داده می‌شوند که روی یک سطح صاف قرار گیرند. هنگام باز نمودن هر وجه تصویر رأس اولیه (بازه‌ای که از آن می‌توان روی سطح صاف حاصل، خط راستی به رأس اولیه رسم نمود) محاسبه می‌شود. اگر این باز کردن وجه‌ها با در نظر گرفتن این بازه‌ی تصویر، با ترتیب خاصی صورت بگیرد در زمانی از $O(n^{۲})$ می‌توان روی سطح حاصل خط راستی از s به t رسم نمود که مسلماً کوتاه‌ترین مسیر اقلیدسی بین s و t می‌باشد.

گونه‌ای دیگر از این مسئله که در عمل کاربرد بسیار زیادی دارد، یافتن کوتاه‌ترین مسیر بین دو نقطه روی سطوح وزن‌دار می‌باشد. در این مسئله به سطح عدد حقیقی مثبتی که نمایانگر هزینه‌ی عبور از واحد طول در آن سطح می‌باشد، نسبت داده می‌شود. هزینه‌ی عبور از هر سطح، حاصل‌ضرب طول قطعه‌ای از مسیر که از آن سطح می‌گذرد در وزن آن سطح می‌باشد. هدف از این مسئله کمینه‌سازی تابع هزینه‌ی کل (مجموع هزینه‌های جزیی) می‌باشد. به عنوان مثال، تعیین مسیری برای جاده‌ی بین دو شهر که از نواحی مختلفی با هزینه‌های جاده‌کشی متفاوت عبور کند یکی از کاربردهای این گونه از مسئله می‌باشد. الگوریتم Mitchell و Papadimitriou بهترین الگوریتم شناخته شده برای این مسئله است که از پیچیدگی زمانی $O(n^{۸}\log n)$ برخوردار است[۴]. با توجه به این پیچیدگی، الگوریتم‌های تقریبی سریع‌تری مورد نیاز است.

الگوریتم‌های تقریبی

الگوریتم‌های تقریبی متنوعی برای یافتن کوتاه‌ترین مسیر در یک تین وجود دارد. حجم بسیار زیاد داده‌های ورودی و عدم نیاز به دقت زیاد و هم‌چنین ماهیت تقریبی خود تین بر اهمیت این الگوریتم‌ها می‌افزاید. سادگی پیاده‌سازی و زمان اجرای کوتاه‌تر آن‌ها از دیگر عوامل توجه به این الگوریتم‌ها است.

دسته‌ای از این الگوریتم‌ها تضمین می‌کند که طول مسیر تقریبی به‌دست آمده از $۱+\epsilon$ برابر طول کوتاه‌ترین مسیر کمتر باشد ($\pi'(s,t) \le (۱+\epsilon)\pi(s,t)$). در این جا ϵ یکی از پارامترهای ورودی الگوریتم می‌باشد و معمولاً با کاهش آن، زمان اجرای الگوریتم افزایش می‌یابد.

دسته‌ی دیگری از این الگوریتم‌ها یک مسیر تقریبی با مقدار خطایی که از قبل مشخص می‌باشد، و بر خلاف الگوریتم‌های قبلی مقدارش در اختیار کاربر نیست، تولید می‌کند. وجود روشی سریع‌تر که بتواند مسیر تقریبی حاصل از این الگوریتم‌ها (و حتی برخی از الگوریتم‌های دسته‌ی قبل) را بهبود بخشد، بسیار مفید است.

الگوریتم Suri و Hershberger یکی از الگوریتم‌هایی است که در حالتی که سطوح بدون وزن فرض شوند، با زمان اجرایی از

[۴] Continuous Dijkstra

پیچیدگی زمانی	فضای مورد نیاز	توضیح	مرجع
$O(n^2)$	$\theta(n)$	کوتاه‌ترین مسیر از یک رأس به تمام رأس‌ها، حالت محدب و غیر محدب، سطوح بدون وزن	Chen و Han [2]
$O(n^8 \log n)$	$O(n^4)$	کوتاه‌ترین مسیر از یک رأس به تمام رأس‌ها، سطوح وزن‌دار	Mitchell و Papadimitriou [4]

جدول ۱: الگوریتم‌های دقیق تعیین کوتاه‌ترین مسیر.

پیچیدگی زمانی	فضای مورد نیاز	توضیح	مرجع
$O(n^5)$	$O(n^4)$	حالت محدب و غیر محدب، سطوح وزن‌دار	Lanthier و بقیه [6]
$O(pn \log pn + np^2)$	—	حالت محدب و غیر محدب، p تعداد نقاط کمکی افزوده شده، سطوح وزن‌دار	Aleksandrov و بقیه [7]
$O(k)$	—	حالت محدب، k تعداد نقاط شکستگی مسیر، سطوح بدون وزن	مهدیان [8]
$O(k) + O(f)$	$O(n)$	حالت محدب و غیر محدب، k تعداد نقاط شکستگی مسیر، f تابع زمانی الگوریتم تقریبی اولیه، سطوح وزن‌دار	الگوریتم پیشنهادی در این مقاله

جدول ۲: الگوریتم‌های تقریبی تعیین کوتاه‌ترین مسیر.

$O(n)$ تقریبی از کوتاه‌ترین مسیر بین دو رأس به‌دست می‌دهد که طولش حداکثر دو برابر طول مسیر بهینه است [۵]. در الگوریتم Lanthier و بقیه، تعدادی رأس کمکی به رأس‌های تین افزوده شده، جستجوی کوتاه‌ترین مسیر به مسیرهایی که از این رأس‌ها می‌گذرند محدود می‌شود [۶].

در این مقاله، روشی را پیشنهاد می‌کنیم و نشان می‌دهیم که با هر بار استفاده از این روش مسیر به‌دست آمده بهتر می‌شود. این روش مسیر تقریبی اولیه‌ای را که می‌تواند از هر الگوریتم تقریبی دل‌خواهی حاصل شده باشد، دریافت می‌کند، تعدادی رأس کمکی به یال‌های اطراف مسیر ورودی اضافه می‌کند و به جستجوی کوتاه‌ترین مسیر بین مسیرهای گذرنده از این رأس‌های کمکی می‌پردازد.

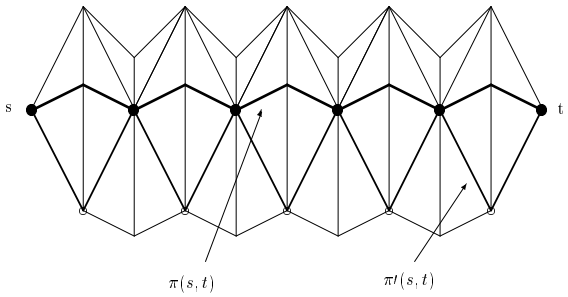
در بخش بعدی به معرفی الگوریتم Lanthier و بقیه می‌پردازیم. نحوه‌ی کار این الگوریتم تا حدودی شبیه به روش پیشنهادی در این مقاله است و از این نظر می‌تواند برای روشن شدن روش پیشنهادی مفید باشد. هم‌چنین از این الگوریتم می‌توان برای به‌دست آوردن مسیر تقریبی اولیه‌ای که روش پیشنهادی لازم دارد، استفاده نمود.

۲ معرفی یک الگوریتم تقریبی

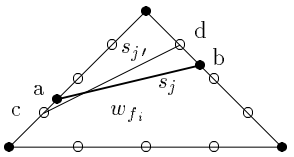
الگوریتم Lanthier و بقیه بر اساس ایجاد یک گراف G متناظر با سطح P (در قالب تین) و افزودن تعدادی رأس کمکی و یال‌های بین آن‌ها به این گراف پایه‌ریزی شده است. رأس‌های G را متناظر با رأس‌های P در نظر می‌گیریم. هر دو رأس از G را با یک یال به هم وصل می‌کنیم اگر بین دو رأس متناظر آن‌ها در P یالی وجود داشته باشد. اگر از الگوریتم دایسترا برای به‌دست آوردن کوتاه‌ترین مسیر بین دو رأس s و t از G (که متناظر با رأس‌های s و t از P هستند) استفاده کنیم، بسته به کیفیت مثلث بندی دقت مسیر متغیر خواهد بود که در بدترین حالت می‌تواند بسیار بد باشد (شکل ۲).

برای جلوگیری از عمل‌کرد بد برنامه و به‌دست آوردن مسیر تقریبی اولیه‌ی بهتر، می‌توان گراف G را با افزودن تعدادی «رأس کمکی[۵]» تکمیل نمود. به این ترتیب که روی هر یال P، m رأس جدید با فاصله‌های متساوی افزوده می‌شود. به طور متناظر این رأس‌ها به گراف G نیز افزوده خواهند شد. هر دو رأسی که روی یک یال P با هم همسایه باشند در گراف G همسایه خواهند بود.

[۵] Steiner Points



شکل ۲: نمونه‌ای از عمل‌کرد بد برنامه.



شکل ۳: قطعه‌ای از کوتاه‌ترین مسیر وزن‌دار (S_j).

هم‌چنین هر دو رأسی که روی سطح f_i از P و روی دو یال متمایز از P واقع شده باشند در گراف G نیز همسایه خواهند بود. در مرحله‌ی بعد با استفاده از یکی از الگوریتم‌های معمولی تعیین کوتاه‌ترین مسیر در گراف‌ها (مانند الگوریتم دایسترا) روی گراف G می‌توان یک مسیر تقریبی به‌دست آورد.

این الگوریتم در عین سادگی پیاده‌سازی، الگوریتمی نسبتاً سریع (از $O(n^5)$) است و برای حالت‌های محدب و غیر محدب و نیز اگر سطوح وزن‌دار باشند نیز جواب‌هایی نزدیک به بهینه به‌دست می‌آورد. شکل ۷ زمان اجرای این الگوریتم را بر حسب تعداد نقاط کمکی به ازای هر یال نشان می‌دهد.

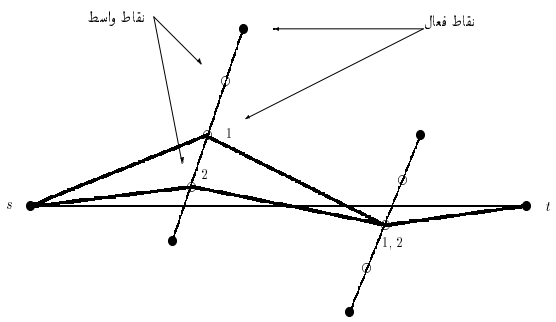
در بخش بعدی به معرفی الگوریتم تقریبی بسیار سریعی می‌پردازیم که می‌توان از آن برای بهبود بخشیدن به مسیر حاصل از این الگوریتم و هر الگوریتم تقریبی دیگری استفاده نمود.

۳ الگوریتم پیشنهادی

مشکل اصلی اکثر الگوریتم‌های تقریبی این است که برای کاهش پیچیدگی زمانی الگوریتم، به صورت حریصانه[۶] عملیات خود را به بخش خاصی از سطح محدود می‌کنند. با توجه به این که برای تعیین ناحیه‌ی تمرکز الگوریتم معمولاً از داده‌های محدود به بازه‌ی خاصی از سطح استفاده می‌شود، این امکان وجود دارد که این انتخاب حریصانه باعث دور شدن الگوریتم از جواب نهایی مسئله شود. از سوی دیگر پردازش کامل سطح مستلزم صرف وقت بسیار زیادی می‌باشد. با توجه به این موضوع، استفاده از الگوریتمی که نخست بر اساس پردازش مقدماتی تمام سطح، مسیر تقریبی اولیه‌ای را به دست آورد و سپس به بهبود مسیر تقریبی یافته شده بپردازد، منطقی به نظر می‌رسد. این موضوع ایده‌ی اصلی الگوریتمی است که در این بخش به تشریح آن می‌پردازیم.

در بخش قبلی الگوریتمی تقریبی معرفی شد که در زمان مناسبی یک تقریب قابل قبول از مسیر با کمترین هزینه را روی سطح وزن دار P به دست می‌دهد. گام بعدی بهبود بخشیدن به این مسیر تقریبی است. این الگوریتم به عنوان ورودی، تین و «مسیر تقریبی اولیه» را دریافت می‌نماید و مسیر بهبود یافته‌ای را به عنوان خروجی تولید می‌کند. برای شروع تعریف‌های زیر را در نظر می‌گیریم:

[۶] Greedy



شکل ۴: رأس‌های واسط و فعال — مسیر ورودی (۱) و خروجی الگوریتم (۲).

رأس‌های همسایه: دو رأس را در تین ورودی همسایه می‌گوییم اگر هر دو، در یک مثلث واقع شده باشند. توجه کنید که اگر ورودی الگوریتم مسیر حاصل از الگوریتم تقریبی معرفی شده در بخش پیش باشد، بهتر است نقاط کمکی افزوده شده در الگوریتم حذف نشوند به طوری که بتوان آن‌ها را به عنوان رأس‌های همسایه در نظر گرفت. این امر هم‌گرایی الگوریتم این بخش را بهبود می‌بخشد.
بعد از یک بار اجرای الگوریتم و ایجاد گراف G دو رأس را همسایه می‌گوییم اگر در گراف G یالی بین آن دو رأس وجود داشته باشد.

رأس فعال: رأس‌های مسیر اولیه‌ای را که به عنوان ورودی به الگوریتم داده می‌شود و نیز رأس‌هایی که در تین ورودی، همسایه‌ی رأس‌های مسیر اولیه محسوب می‌شوند، رأس‌های فعال می‌نامیم.

رأس واسط: نقطه‌ی وسط پاره خط واصل بین هر رأس از مسیر اولیه‌ی ورودی و هر یک از همسایگان آن را رأس واسط می‌نامیم (شکل ۴).

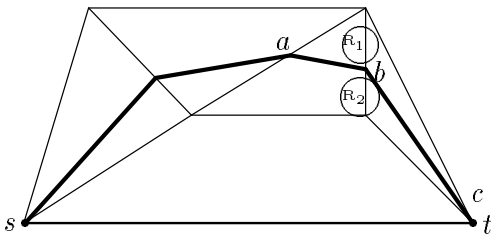
از این پس، رأس‌های گراف G را به رأس‌های فعال، رأس‌های واسط و یال‌های بین آن‌ها محدود می‌کنیم. رأس‌های مسیر ورودی را به ترتیب $s_1 = s$، s_2، ... و $s_k = t$ می‌نامیم. رأس s_i از گراف G (و همین طور رأس‌های همسایه‌ی s_i و رأس‌های واسط مجاور s_i) به رأس‌های s_{i-1} و s_{i+1} (در صورت وجود) و نیز به همسایه‌های s_{i-1} و s_{i+1} و هم چنین به رأس‌های واسط مجاور آن دو وصل می‌شود (و بنا بر این وزن یال بین آن‌ها حاصل ضرب فاصله‌ی اقلیدسی بین آن‌ها در وزن وجه مشترکشان می‌باشد). بقیه‌ی رأس‌ها جدا از هم محسوب می‌شوند. با توجه به این امر به سادگی مشاهده می‌شود که:

لم ۱. حداکثر درجه‌ی هر رأس در گراف G، ۱۲ است.

در این الگوریتم نیز از یکی از الگوریتم‌های تعیین کوتاه‌ترین مسیر (مانند دایسترا) برای تعیین مسیر جدید بین s و t استفاده می‌شود.

از این الگوریتم می‌توان چندین بار روی یک مسیر استفاده نمود. فقط باید توجه داشت که برای اجرای مجدد الگوریتم، باید رأس‌های فعال و رأس‌های واسط را روز آمد کرد. به این معنی که بدون حذف هیچ یک از رأس‌های افزوده شده در الگوریتم، یک‌بار دیگر رأس‌های فعال را مشخص می‌کنیم و رأس‌های واسط نیز افزوده می‌شوند. سپس همه رأس‌ها به‌جز رأس‌های واسط و فعال از گراف حذف می‌شوند و الگوریتم تعیین کوتاه‌ترین مسیر دوباره اجرا می‌گردد.
در حالتی که سطوح وزن‌دار نباشند، می‌توان نشان داد که:

قضیه ۱. با اجرای متوالی الگوریتم فوق، تا رسیدن به یک جواب کمینه‌ی محلی (مسیری که در بین تمام مسیرهای گذرنده از



شکل ۵: قسمتی از مسیر اولیه و نحوه‌ی تغییر وضعیت راس‌ها.

سطوحی که مسیر اولیه از آن‌ها می گذرد و هم‌چنین سطح‌های همسایه‌ی این سطوح، کم‌ترین طول را داشته باشد) طول مسیر کاهش می‌یابد.

اثبات. وجه‌هایی را که مسیر اولیه‌ی ورودی الگوریتم از آن‌ها عبور کرده است، در نظر بگیرید. این وجه‌ها را طوری نسبت به هم دوران می‌دهیم که همگی در یک صفحه قرار گیرند. برای این کار از وجه اولیه‌ای که مسیر ورودی از آن عبور کرده شروع به حرکت می‌کنیم و با هر عبور از مرز بین دو وجه، وجه جدید را طوری دوران می‌دهیم که با وجه قبلی روی یک صفحه قرار بگیرد. روی این صفحه خط راستی را که s و t را به هم وصل می‌کند، در نظر می‌گیریم (شکل ۵). اگر تمامی نقاط مسیر اولیه روی این خط راست قرار گرفته باشند، با توجه به این که این مسیر بین تمام مسیرهای گذرنده از این وجه‌ها کم‌ترین طول را دارد، مسلماً به یک مسیر بهینه‌ی محلی رسیده‌ایم. در غیر این صورت سه رأس فعال متوالی a، b و c از مسیر اولیه را در نظر می‌گیریم که روی یک خط راست واقع نباشند. اگر بعد از چند بار اجرای پیاپی الگوریتم طول مسیر کاهش نیابد به این معنی است که نقاط a، b و c همواره جزیی از مسیر بوده‌اند. در این حالت بعد از چند بار اجرای الگوریتم نقاط واسط به نواحی R_1 و R_2 در نزدیکی دو طرف مسیر خواهند رسید (شکل ۵). بدیهی است (با توجه به این که a، b و c روی یک خط راست قرار ندارند) که مسیرهایی که از یکی از این دو ناحیه می‌گذرند از مسیر گذرنده از a، b و c کوتاه‌تر است که همین باعث رسیدن به مسیری با طول کم‌تر خواهد شد. □

توجه کنید که چون در حین اجرای الگوریتم ممکن است وجه‌هایی که مسیر از آن‌ها می‌گذرد تغییر یابد، این الگوریتم می‌تواند به مسیر کمینه در بخش وسیعی از سطح ورودی هم‌گرا شود. در واقع می‌توان سطح اولیه را به صورت مجموعه‌ای از نواحی (که ممکن است با هم اشتراک داشته باشند) در نظر گرفت که این الگوریتم با شروع به کار روی هر مسیری در هر کدام از این نواحی به بهترین مسیر آن ناحیه هم‌گرا می‌شود.

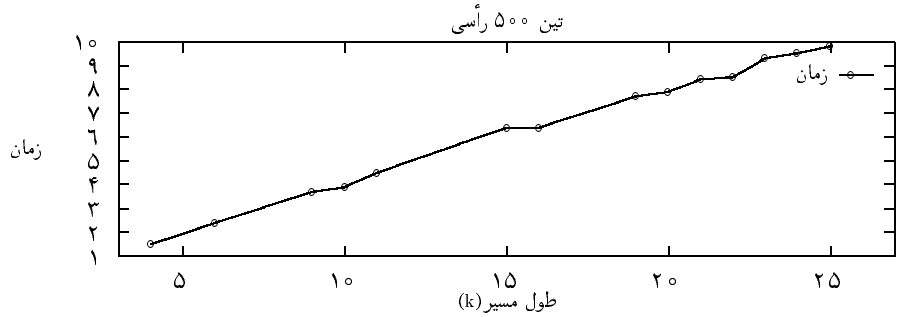
در واقع اهمیت گام نخست در تعیین مسیر اولیه در این است که خوب بودن مسیر تقریبی اولیه موجب افزایش احتمال قرار گرفتن مسیر در ناحیه‌ی مربوط به بهترین مسیر می‌شود و الگوریتم پیشنهادی با قرار گرفتن در این ناحیه به سرعت به جواب بهینه هم‌گرا می‌شود.

لازم به توضیح است که حالتی که برای اثبات قضیه‌ی فوق در نظر گرفته شده است در عمل بسیار به‌ندرت اتفاق می‌افتد و همان‌گونه که در نمودار درصد خطای الگوریتم دیده می‌شود (شکل ۸)، این الگوریتم به سرعت به جواب بهینه هم‌گرا می‌شود.

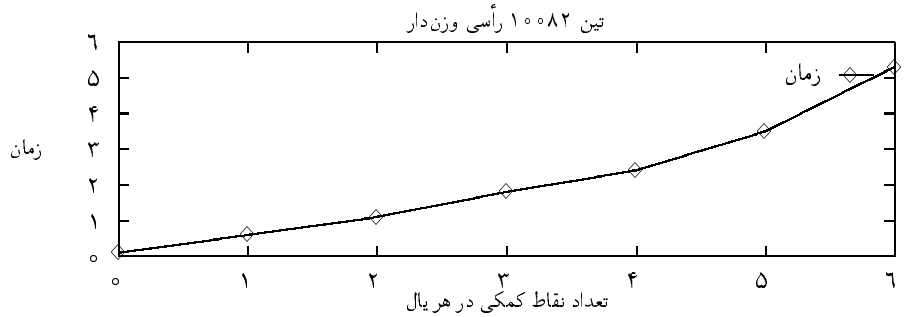
۴ تحلیل الگوریتم پیشنهادی

اگر k را تعداد رأس‌های مسیر اولیه‌ی ورودی در نظر بگیریم به وضوح تعداد رأس‌های G از $O(k)$ خواهد بود. همین طور با توجه به لم ۱ تعداد یال‌های G نیز از $O(k)$ می‌باشد. با توجه به این امر و با توجه به ساختار خاص گراف G اگر برای تعیین کوتاه‌ترین مسیر از الگوریتم دایسترا با استفاده از هیپ فیبوناچی[۷] استفاده نماییم، هر بار اجرای الگوریتم بالا از $O(k)$ خواهد بود. شکل ٦، زمان اجرای این الگوریتم را بر حسب طول مسیر اولیه برای یک تین با ۵۰۰ رأس نمایش می‌دهد. طبیعی است که زمان اجرای این الگوریتم برای سطوح وزن‌دار و سطوح بدون وزن مرتبه‌ی یکسانی دارد.

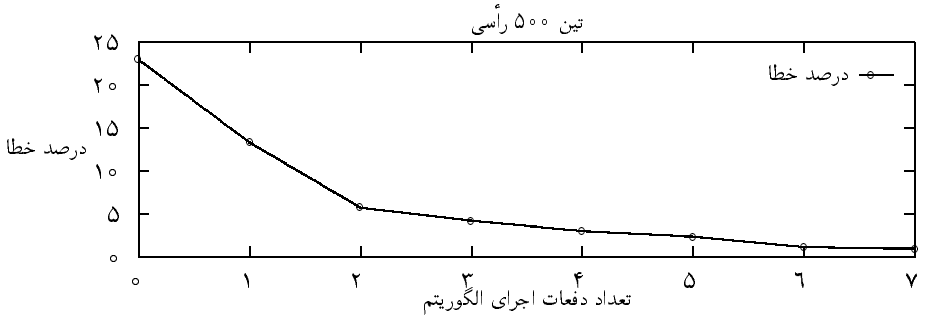
[۷] Fibonacci Heap



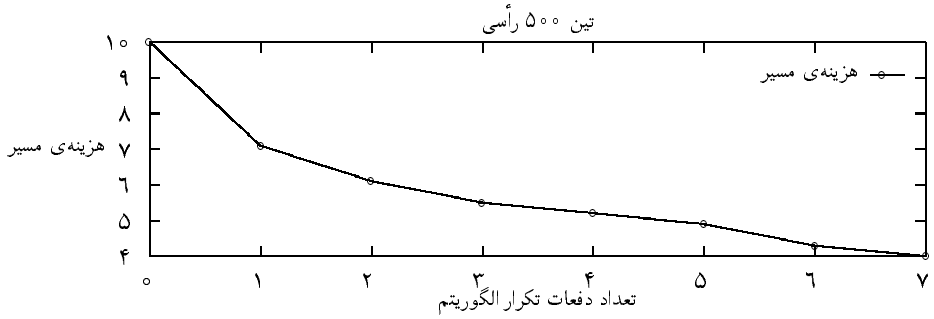
شکل ٦: زمان اجرای الگوریتم پیشنهادی.



شکل ۷: زمان اجرای الگوریتم Lanthier روی سطح وزن‌دار.



شکل ۸: درصد خطای الگوریتم در حالت سطوح بدون وزن.



شکل ۹: هزینه‌ی مسیر تقریبی حاصل از الگوریتم روی یک سطح وزن‌دار.

۵ تحلیل خروجی الگوریتم

با وجود این که قضیه‌ی ۱ در حالتی که سطوح ورودی بدون وزن هستند ثابت شده است، نتایج حاصل از اجرای این الگوریتم برای سطوح وزن‌دار نشان می‌دهد که این الگوریتم در این حالت هم از هم‌گرایی بسیار بالایی برخوردار است. شکل ۸ نشان دهنده‌ی رابطه‌ی درصد خطای مسیر حاصل از الگوریتم با تعداد دفعات اجرای الگوریتم، برای حالت سطوح بدون وزن است. شکل ۹ نیز میزان هم‌گرایی مسیر حاصل از الگوریتم را برای حالت سطوح وزن دار نمایش می‌دهد. در هر دوی این نمودارها در گام نخست فقط از دو رأس کمکی در هر یال استفاده شده است.

۶ نتیجه‌گیری

در این مقاله به معرفی الگوریتمی پرداختیم که مسیر اولیه‌ای با k رأس را به عنوان ورودی دریافت می‌کند و با زمان اجرایی از $O(k)$ مسیر بهبود یافته‌ای را به عنوان خروجی تولید می‌نماید. همان گونه که نتایج آماری حاصل از عمل کرد برنامه نشان می‌دهد، این برنامه در حالت سطوح وزن‌دار نیز به خوبی عمل می‌کند. بهترین الگوریتم دقیق شناخته شده برای مسئله‌ی کوتاه‌ترین مسیر در حالت وزن‌دار، زمان اجرایی از $O(n^8 \log n)$ دارد که با توجه به عدم نیاز به دقت بالا در موارد عملی، نشان دهنده‌ی کارآیی خوب الگوریتم پیشنهادی است.

تشکر و قدردانی: از آقای فؤاد دبیری که قسمت‌هایی از الگوریتم را پیاده‌سازی کردند، تشکر می‌نماییم.

مراجع

[1] Yie-Huei Hwang, Ruei-Chuan Chang and Hung-Yi Tu, *"Finding all shortest path edge sequences on a convex polyhedron"*, LNCS 382, Algorithms and Data Structures Workshop, Ottawa, Canada. 251-266 (Aug. 1989).

[2] J. Chen, Y. Han, *"Shortest Paths on a Polyhedron"*, International Journal of Computational Geometry & Application, Vol. 6, No. 2 (1996), 127-144.

[3] M. Sharir and A. Schorr, *"On shortest paths in polyhedral spaces"*, SIAM Journal of Computing, 15 (1986), pp. 193-215.

[4] J.S.B. Mithcell and C.H. Papadimitriou, *"The Weighted Region Problem: Finding Shortest Paths Through a Weighted Planar Subdivision"*, Journal of the ACM, 38, January 1991, pp.18-73.

[5] J. Hershberger and S. Suri, *"Practical Methods for Approximating Shortest Paths on a Convex Polytope in R^3"*, SODA'95, 1995.

[6] M. Lanthier, A. Maheshwari and J. Sack, *"Approximating Weighted Shortest Paths on Polyhedral Surfaces"*, Tech. Report TR-96-32, School of Computer Science, Carleton University, Dec. 1996.

[7] L Aleksandrov, M. Lanthier and A. Maheshwari, *"An ϵ-Approximation Algorithm for Weighted Shortest Paths on Polyhedral Surfaces"*, SWAT 98, Sweden, July 1998.

[۸] محمد مهدیان، *«کوتاه‌ترین مسیر در سطوح نامنظم مثلث بندی شده»*، رساله‌ی کارشناسی، دانشکده‌ی مهندسی کامپیوتر، دانشگاه صنعتی شریف، آبان ماه ۱۳۷۶.